ETH Zürich
Communication Systems Group
Lecturers: Prof. Dr. Bernhard Plattner
Dr. P. Georgopoulos, Dr. B. Ager
Dr. M. Happe, Dr. K. Hummel
TA: Vasileios Kotronis

HS 2014
Advanced Topics in Communication Networks
Tasks of **Exercise 6**
Assigned: 10 December 2014
Debriefing: 17 December 2014

Exercise 6

Introduction

In this exercise you will become familiar with deploying and running OpenFlow-based [9] experiments on a real OpenFlow testbed distributed across Europe; the OFELIA testbed [5]. In particular, you will replicate the setup of exercise 1 ("Building a simple load balancer on top of the POX OpenFlow controller" [1]) over the OFELIA testbed island at ETHZ [2]. You can also use any other island that belongs to the OFELIA testbed, but keep in mind that since the OFELIA project has ended, the testbed is a best-effort facility now, in terms of administration and maintenance. The ETHZ island offers switches running OpenFlow v1.0 [10] that are sliced via FlowVisor and are presented as virtualized network resources (ports, flowspaces). In addition, the ETHZ island offers Virtual Machines (VMs) running on top of physical XEN servers. These VMs can be used as end-hosts for any kind of experimentation, including the deployment of OpenFlow controllers.

Optional Prerequisite

As a side note, in order to connect to the OFELIA control network you need to update your network and DNS settings on your computer (see instructions later). If you are reluctant to do this on your every-day OS, we recommend using a Linux VM on top of your regular OS via a hypervisor (e.g., VirtualBox), so as not having to change any network or DNS configuration on your system.

Tasks

Figure 1 depicts the high lever user work flow in order to run experiments on the OFELIA OpenFlow testbed. This might seem a bit daunting at first, but it is worth the effort when you realise that you will be able to run OpenFlow experiments on a real testbed running across Europe (although you are asked for simplicity to run your experiment on only one OpenFlow island, i.e., the ETHZ one, nothing precludes you from running federated tests).

In particular, the tasks that you will have to implement for this exercise are the following :

1. Learn about OFELIA. The suggested information sources are the following:
   (a) ATCN lecture on SDN Testbeds and Experimentation delivered on 8/12/2014 [8].
   (b) Official OFELIA project website [5].
   (c) OFELIA user manual and documentation wiki [4].
   (d) OFELIA tutorial [3].
   Note: This tutorial plus the wiki information [4] will be your main guides for completing this exercise.

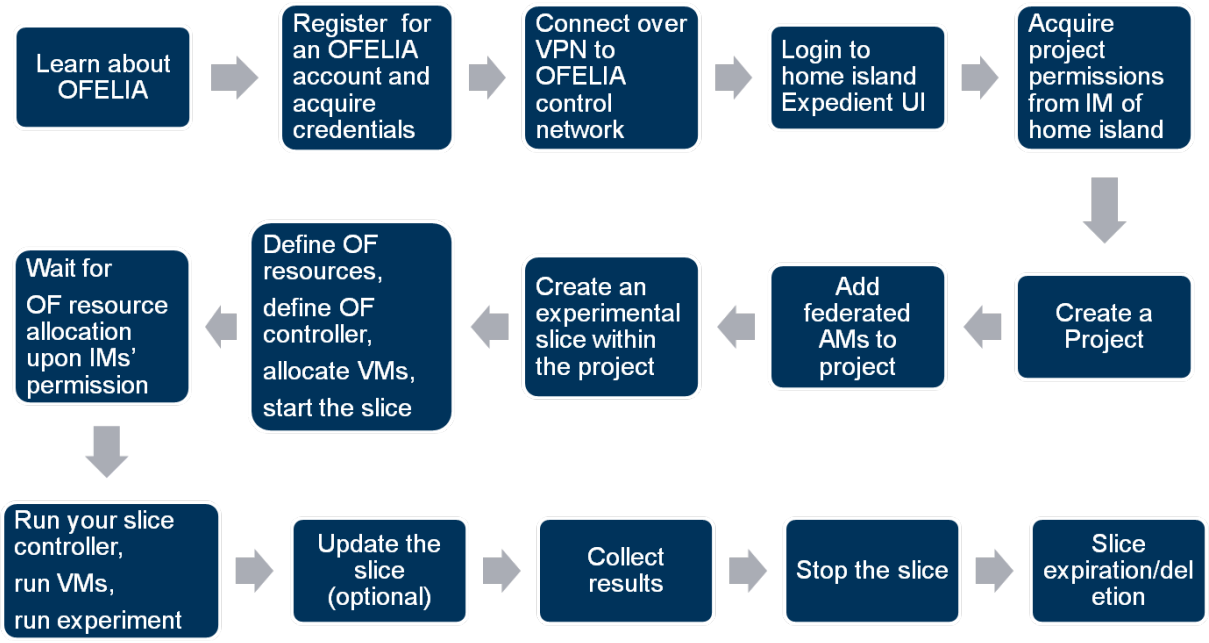
Learn about OFELIA
Register for an OFELIA account and acquire credentials
Connect over VPN to OFELIA control network
Login to home island Expedient UI
Acquire project permissions from IM of home island
Create a Project
Add federated AMs to project
Create an experimental slice within the project
Define OF resources, define OF controller, allocate VMs, start the slice
Wait for OF resource allocation upon IMs' permission
Run your slice controller, run VMs, run experiment
Update the slice (optional)
Collect results
Stop the slice
Slice expiration/deletion

Figure 1: OFELIA User Work Flow

(e) ETHZ island description [2].

(f) OFELIA whitepaper [11].

2. Register to the OFELIA testbed in order to acquire a login account and the matching credentials (step 2.1 of the OFELIA tutorial [3]).
ATTENTION: The registration URL described in [3] has changed since it was last written. The updated registration URL is:
`https://register.fp7-ofelia.eu/registration/register`

3. Login to OFELIA and setup your VPN connection to access the OFELIA testbed (step 2.2 of the OFELIA tutorial [3]).
ATTENTION: The login URL described in [3] has changed since it was last written. The updated URL is:
`https://register.fp7-ofelia.eu/login/login`.

4. Connect over VPN to the OFELIA control network (step 2.2 of the OFELIA tutorial [3]).

5. Login to the OFELIA Control Framework (step 2.3 of the OFELIA tutorial [3]).

6. Create a new project, request and acquire the necessary permissions from the island manager of your home island (step 3.1 of the OFELIA tutorial [3]).

7. Add the required Aggregate Managers (AMs) to your project (e.g., ETHZ island Virtualization AM, ETHZ island Openflow AM).

8. Create an experimental slice within your project (step 3.2 of the OFELIA tutorial [3]). Make sure you also add the needed AMs to your slice, in order to be able to access the associated resources.

9. Set the OpenFlow controller for your OpenFlow network (step 3.2 of the OFELIA tutorial [3]). You can use the IP of the OFELIA OpenVPN tunnel interface on your computer (home VM), or the IP of any other created VM within the island, which belongs to your slice (see later steps on how to create VMs using the virtualization AM interface). This step is optional at this point.

10. Set a request for OpenFlow resources (step 3.3 of the OFELIA tutorial [3]). In the current version of the user interface, you have the option to choose between "Simple" and "Advanced" resource

allocation processes. In "Simple" mode, you simply specify the ports and datapath ids that you want to use from the testbed topology (textually or graphically), and then an available VLAN from the island will automatically be assigned to the requested flowspace. In "Advanced" mode, you again specify the ports and datapath ids that you want to use from the testbed topology as a first step, but as a second step you can also specify your own flowspace using tables of ranges (e.g., "I want the IPs between 192.168.56.1 and 192.168.56.4"). In the end, an available VLAN by the island manager will also be assigned to the requested flowspace. If you are for example using the ETHZ island to perform the exercise's tasks, you can ask for the following resources via the OpenFlow AM interface:

- Switch 02:00:00:00:00:00:00:02, port 13: this port faces the interface eth2 of the VM server ofelia-ethz3.
- Switch 02:00:00:00:00:00:00:02, port 14: this port faces the interface eth2 of the VM server ofelia-ethz2.
  ATTENTION: due to a bug in the backend database, this port appears to connect to interface eth1 of ofelia-ethz2, while in reality it connects to interface eth2. The full (correct) topology can also be found at [2].

The above switch ports, facing two data plane interfaces (in this case eth2) of two VM servers of the ETHZ island, should provide all the network resources you would require to proceed with your task. Remember that for re-implementing exercise 1 you will need only one switch. You can allocate the "clients" of the exercise on one VM XEN server and the "servers" of the exercise on another VM XEN server, as both XEN servers are connected to the aforementioned switch. One port of the switch will face the virtual kernel bridge associated with the eth2 interface of the first VM server where "clients" reside, while the other port will face the virtual kernel bridge associated with the eth2 interface of the second VM server where "servers" reside. We mention the term "kernel bridge" because this is in reality the way that multiple virtual interfaces are multiplexed and connected to the physical interface of the server. However, you do not have to deal with this detail as experimenters.

Reminder: these OpenFlow resources will be booked but are not yet requested and allocated at this stage (see later steps for those processes).

11. Create the VMs you need (step 3.4 of the OFELIA tutorial [3]). According to the setup of exercise 1, you can allocate 3 VMs on ofelia-ethz2, representing "clients", and 3 VMs on ofelia-ethz3, representing "servers". Please use the minimum RAM requirement for VMs, which is 128 MBytes. Also, please do NOT include underscores in the VM's name, since this will lead to the failure of the creation of your VMs. After creation, you can start, reboot, stop or delete your VMs at any time. Each VM will automatically get a control IP on interface eth0; you can use this IP to ssh into the VM, as long as you are connected to the OFELIA control network over VPN.

    Note: Besides the client and server VMs, you are advised to create a VM that will host your OpenFlow controller (e.g., POX). After you create this VM, please set the correct controller IP for the control of the OpenFlow resources (see "Network resources" part of the slice management webpage).

    ATTENTION: We highly recommend to use a small number of VMs (e.g., 2-3 clients, 2-3 servers and one controller) for your experiment with the load balancer. The testbed resources are not infinite, thus heavy usage by multiple experimenters may result in denying the creation of new VMs. In this case, please send an email to the TA for further consulting.

12. Start your experiment/slice (step 4.1 of the OFELIA tutorial [3]).

    ATTENTION: Until you receive the flowspace approval by the island manager, your flowspace has not yet been granted and cannot be used. Only granted flowspace is known to the FlowVisor that handles the slicing of the OFELIA OpenFlow resources. This will occur when the island

manager approves your flowspace. You will be notified by email and will notice the change in the slice management page after the approval is successful.

13. Set the data plane interfaces on your client and server VMs so as to run your experiment (step 4.2 of the OFELIA tutorial [3]).
ATTENTION: Please check the VLAN tag that has been assigned from the island manager to your experiment/flowspace before using the interfaces on the VMs. For example, if you have been assigned VLAN X, you have to configure the eth2 interface of all your VMs (on server ofelia-ethz2 and ofelia-ethz3 servers) as eth2.X. Since you will need root access to install the vlan package and configure the virtual interfaces make sure you change your permissions using "su", with password "openflow" while being ssh-ed to the respective VM.

14. At this point, your setup should look like the one in Figure 2, assuming that you are using the ETHZ island to run your experiments. In case not enough resources are available at ETHZ island, you are advised to use any other of the rest of the OFELIA islands. For example, the i2CAT island in Spain and and CreateNet in Italy are considered quite stable for running reliable experiments.

15. Now you can test the OpenFlow load balancer application that you wrote for exercise 1 (the code should be portable). To achieve this, you can simply download POX [6, 7] on your OpenFlow controller VM (which you either have to create in one of the island servers—preferably not the ones used for the exercises' clients or servers— or run it locally on your computer) and run the

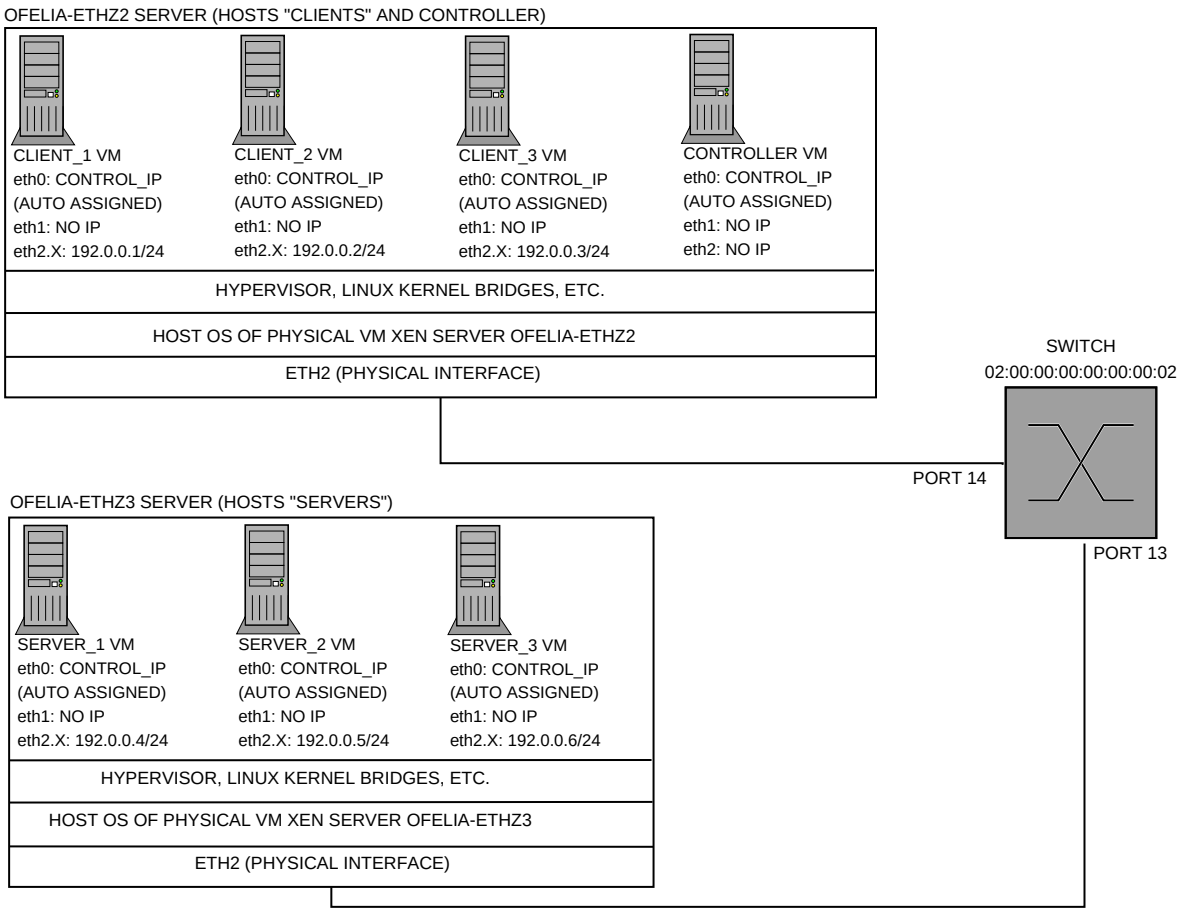

Figure 2: Sample setup for running exercise 1 on the OFELIA ETHZ testbed island. In case you use other islands, the setup should be adjusted accordingly. Only the data plane connectivity (e.g., over the eth2 interface of the servers) is depicted for simplicity. Control plane connectivity (e.g., for accessing the VMs and controlling the switches) is carried out out-of-band. Control IPs are automatically assigned by the framework and are unique per VM control interface.

OpenFlow controller as specified in exercise 1 (you may of course adjust the IPs you are using for the clients and servers). The python script you developed for exercise 1 should suffice.

16. After the experiment is concluded, make sure you clean up your slice and free resources for other experimenters (step 4.3 of the OFELIA tutorial [3]). That is, make sure that you have stopped your slice (i.e., deallocated your flowspace) and that you have stopped and deleted your VMs.

Submission and Debriefing

There is no submission required for this exercise. Its purpose is to encourage you to start experimenting on real-world SDN testbeds using your portable OpenFlow code. Also, you will see that using an open interface to the network resources is of great value. This means that you can interface both with the OpenVswitch used in Mininet and a real OpenFlow switch (used in OFELIA testbeds) without having to change your controller's code.

Debriefing

All students are requested to attend the debriefing session on December 17, where a short discussion and recap of the exercise will take place. At the same day a wrap-up of all the exercises of the course will occur. You will have the opportunity to provide feeback to the TA, participate in the general discussion, form Q&A, recap what you have learned and suggest further improvements pertaining to the exercises and the course in general.

Asking for help

For any issues you may encounter during the exercise please ask the teaching assistant V. Kotronis during the exercise sessions (13:00-15:00) at ETZ G91 on 10/12, and/or 17/12, or send him an email to:

`vkotroni@tik.ee.ethz.ch`

Enjoy!

References

[1] Exercise 1: Building a simple load balancer on top of the POX OpenFlow controller. http://www.csg.ethz.ch/education/lectures/ATCN/hs2014/exercises/exercise1_assignment.pdf.

[2] OFELIA Testbed Zurich (ETHZ island). https://alpha.fp7-ofelia.eu/doc/index.php/Testbed_Zurich.

[3] OFELIA tutorial. http://www.fp7-ofelia.eu/assets/Uploads/OFELIA-Tutorial.pdf.

[4] OFELIA user manual and documentation wiki. https://alpha.fp7-ofelia.eu/doc/index.php/Main_Page.

[5] Official OFELIA project website. http://www.fp7-ofelia.eu/.

[6] POX Git Repository. https://github.com/noxrepo/pox.

[7] POX Wiki. https://openflow.stanford.edu/display/ONL/POX+Wiki.

[8] Vasileios Kotronis. ATCN Lecture: SDN Testbeds and Experimentation. http://www.csg.ethz.ch/education/lectures/ATCN/hs2014/material/SDN_Testbeds.pdf.

[9] Nick McKeown, Tom Anderson, Hari Balakrishnan, Guru Parulkar, Larry Peterson, Jennifer Rexford, Scott Shenker, and Jonathan Turner. OpenFlow: enabling innovation in campus networks. *SIGCOMM CCR*, 38(2):69–74, 2008.

[10] Open Networking Foundation. OpenFlow Switch Specification, v1.0.0. https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.0.0.pdf.

[11] Marc Suñé, Leonardo Bergesio, Hagen Woesner, Tom Rothe, Andreas Köpsel, Didier Colle, Bart Puype, Dimitra Simeonidou, Reza Nejabati, Mayur Channegowda, et al. Design and Implementation of the OFELIA FP7 Facility: the European OpenFlow Testbed. *Computer Networks*, 61:132–150, 2014.